

# Branch-price-and-cut for vehicle routing

Guy Desaulniers

Professor, Polytechnique Montréal, Canada  
Director, GERAD, Canada

VeRoLog PhD School 2018  
Cagliari, Italy, June 2, 2018



POLYTECHNIQUE  
MONTRÉAL

LE GÉNIE  
EN PREMIÈRE CLASSE



GROUPE D'ÉTUDES ET DE RECHERCHE  
EN ANALYSE DES DÉCISIONS

# Outline

- 1 VRPTW definition
- 2 Mathematical formulations
  - Arc-flow formulation
  - Dantzig-Wolfe decomposition
  - Path-flow formulation
- 3 Branch-price-and-cut
  - Master problem
  - Pricing problem
  - Linear relaxation computational results
  - Branching
  - Cutting
  - Using upper bounds
- 4 Computational results (VRPTW and CVRP)

# Outline

- 1 VRPTW definition
- 2 Mathematical formulations
  - Arc-flow formulation
  - Dantzig-Wolfe decomposition
  - Path-flow formulation
- 3 Branch-price-and-cut
  - Master problem
  - Pricing problem
  - Linear relaxation computational results
  - Branching
  - Cutting
  - Using upper bounds
- 4 Computational results (VRPTW and CVRP)

# Vehicle routing

## Broad definition

Vehicle routing consists of determining **least-cost vehicle routes** to accomplish a set of **tasks** while satisfying side constraints

Task examples: a delivery or a pickup-and-delivery request

## Many VRP variants

- Focus on variants where each task must be performed **exactly once by a single vehicle**
- The vehicle routing problem with time windows (**VRPTW**) is used here as an illustrative example

## Elementarity requirements

A route cannot cover the same task more than once

# Vehicle routing

## Broad definition

Vehicle routing consists of determining **least-cost vehicle routes** to accomplish a set of **tasks** while satisfying side constraints

Task examples: a delivery or a pickup-and-delivery request

## Many VRP variants

- Focus on variants where each task must be performed **exactly once by a single vehicle**
- The vehicle routing problem with time windows (**VRPTW**) is used here as an illustrative example

## Elementarity requirements

A route cannot cover the same task more than once

# Vehicle routing

## Broad definition

Vehicle routing consists of determining **least-cost vehicle routes** to accomplish a set of **tasks** while satisfying side constraints

Task examples: a delivery or a pickup-and-delivery request

## Many VRP variants

- Focus on variants where each task must be performed **exactly once by a single vehicle**
- The vehicle routing problem with time windows (**VRPTW**) is used here as an illustrative example

## Elementarity requirements

A route cannot cover the same task more than once

# Capacitated vehicle routing problem (CVRP)

## Definition

- **Given**
  - unlimited number of identical vehicles with a given capacity, housed in a single depot
  - set of customers with known demands (all pickups or all deliveries)
- **Find** vehicle routes such that
  - all customer demands are satisfied
  - each customer is visited by a single vehicle
  - each route starts and ends at the depot
  - each route satisfies vehicle capacity
  - total cost (distance) is minimized

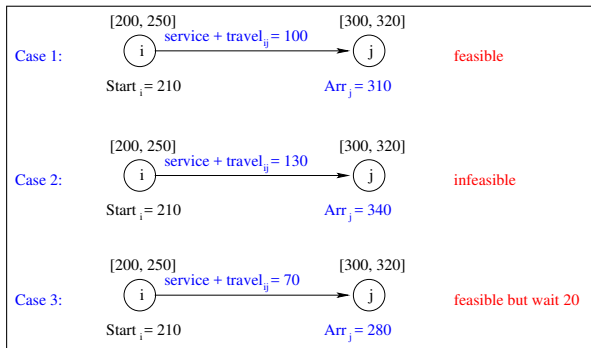
# Vehicle routing problem with time windows (VRPTW)

## Definition

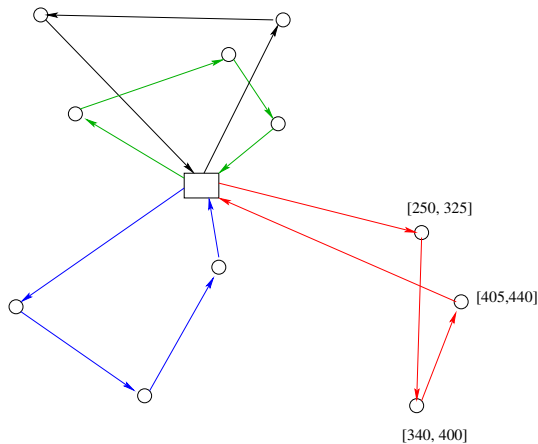
- **Given**
  - unlimited number of identical vehicles with a given capacity, housed in a single depot
  - set of customers with known demands
  - travel and service times
  - a time window for each customer
- **Find** vehicle routes such that
  - all customer demands are met
  - each customer is visited by a single vehicle
  - each route starts and ends at the depot
  - each route satisfies vehicle capacity and time windows
  - total cost (distance) is minimized

# Time windows

- At every customer, service must **start** within its time window
- Hard** time windows
  - Infeasible if upper bound is exceeded
  - Must wait until lower bound if vehicle arrives too early



# Example of a solution



# Outline

- 1 VRPTW definition
- 2 Mathematical formulations
  - Arc-flow formulation
  - Dantzig-Wolfe decomposition
  - Path-flow formulation
- 3 Branch-price-and-cut
  - Master problem
  - Pricing problem
  - Linear relaxation computational results
  - Branching
  - Cutting
  - Using upper bounds
- 4 Computational results (VRPTW and CVRP)

# Notation

**Customer** set  $N = \{1, \dots, n\}$

- demand  $q_i$
- time window  $[a_i, b_i]$

**Depot** represented by two nodes  $\{o, d\}$

- no demand:  $q_o = q_d = 0$
- planning horizon:  $[a_o, b_o] = [a_d, b_d]$

**Node** set  $\bar{N} = N \cup \{o, d\}$

**Vehicle** set  $K$ 

- $|K|$  is sufficiently large to be unrestrictive
- capacity  $Q$

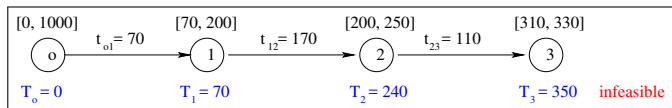
**Arc** set  $A \subset \bar{N} \times \bar{N}$ 

- travel time  $t_{ij}$  (including service time at  $i$ )
- travel cost  $c_{ij}$
- $(i, j) \in A$  if  $q_i + q_j \leq Q$  and  $a_i + t_{ij} \leq b_j$

## Route feasibility

**Network**  $G = (\bar{N}, A)$

- **Every feasible route** corresponds to an  $o$ - $d$  path in  $G$
- However, **NOT** all feasible  $o$ - $d$  paths in  $G$  correspond to a feasible route



## Decision variables

**Arc-flow** variables for all  $(i, j) \in A$ ,  $k \in K$

$$X_{ij}^k = \begin{cases} 1 & \text{if vehicle } k \text{ uses arc } (i, j) \\ 0 & \text{otherwise} \end{cases}$$

**Time** variables for all  $i \in \bar{N}$ ,  $k \in K$

$T_i^k$  = start of service of vehicle  $k$  at node  $i$

## Remark

$T_i^k$  is irrelevant if vehicle  $k$  does not visit node  $i$

# Formulation (1)

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} X_{ij}^k \quad (1) \quad \text{total cost}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{(i,j) \in A} X_{ij}^k = 1, \quad \forall i \in N \quad (2) \quad \text{visit each customer}$$

$$\sum_{(o,j) \in A} X_{oj}^k = \sum_{(i,d) \in A} X_{id}^k = 1, \quad \forall k \in K \quad (3) \quad \text{from and to depot}$$

$$\sum_{(i,j) \in A} X_{ij}^k - \sum_{(j,i) \in A} X_{ji}^k = 0, \quad \forall k \in K, i \in N \quad (4) \quad \text{flow conservation}$$

# Formulation (2)

$$\sum_{(i,j) \in A} q_i X_{ij}^k \leq Q, \quad \forall k \in K \quad (5) \quad \text{vehicle capacity}$$

$$X_{ij}^k (T_i^k + t_{ij} - T_j^k) \leq 0, \quad \forall k \in K, (i,j) \in A \quad (6) \quad \text{time computation}$$

$$a_i \leq T_i^k \leq b_i, \quad \forall k \in K, i \in \bar{N} \quad (7) \quad \text{time windows}$$

$$X_{ij}^k \in \{0, 1\}, \quad \forall k \in K, (i,j) \in A \quad (8) \quad \text{binary requirements}$$

# Remarks

## Set partitioning vs set covering

When costs and travel times satisfy the triangle inequality, **set partitioning** constraints (2)

$$\sum_{k \in K} \sum_{(i,j) \in A} x_{ij}^k = 1, \quad \forall i \in N$$

can be replaced by **set covering** constraints

$$\sum_{k \in K} \sum_{(i,j) \in A} x_{ij}^k \geq 1, \quad \forall i \in N.$$

- Reduces the dual space in half
- Can speed up solution process

## Linearization

### Constraints (6)

$$X_{ij}^k (T_i^k + t_{ij} - T_j^k) \leq 0, \quad \forall k \in K, (i, j) \in A$$

are **nonlinear**, but can be linearized as

$$T_i^k + t_{ij} - T_j^k \leq M_{ij}(1 - X_{ij}^k), \quad \forall k \in K, (i, j) \in A$$

where  $M_{ij} = b_i + t_{ij} - a_j$  is a large constant

Yields a **very weak** linear relaxation

# Dantzig-Wolfe decomposition (in a few slides)

- Consider the integer linear program (**compact formulation**):

$$\begin{aligned}
 \min \quad & \mathbf{c}^\top \mathbf{x} \\
 \text{s.t.} \quad & \mathbf{Ax} \geq \mathbf{b} \\
 & \mathbf{Dx} \geq \mathbf{d} \\
 & \mathbf{x} \geq \mathbf{0} \\
 & \mathbf{x} \in \mathbb{Z}^n
 \end{aligned} \tag{9}$$

- Define the domains:

$$\begin{aligned}
 \mathcal{A} &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Ax} \geq \mathbf{b}\} && \neq \emptyset \\
 \mathcal{D} &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Dx} \geq \mathbf{d}, \mathbf{x} \geq \mathbf{0}\} && \neq \emptyset \\
 \mathcal{X} &= \mathcal{D} \cap \mathbb{Z}^n
 \end{aligned} \tag{10}$$

# Dantzig-Wolfe decomposition (in a few slides)

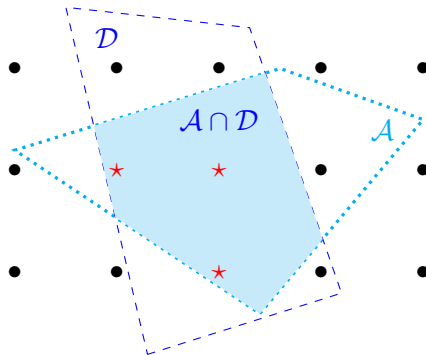
- Consider the integer linear program (**compact formulation**):

$$\begin{aligned}
 \min \quad & \mathbf{c}^\top \mathbf{x} \\
 \text{s.t.} \quad & \mathbf{Ax} \geq \mathbf{b} \\
 & \mathbf{Dx} \geq \mathbf{d} \\
 & \mathbf{x} \geq \mathbf{0} \\
 & \mathbf{x} \in \mathbb{Z}^n
 \end{aligned} \tag{9}$$

- Define the domains:

$$\begin{aligned}
 \mathcal{A} &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Ax} \geq \mathbf{b}\} && \neq \emptyset \\
 \mathcal{D} &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Dx} \geq \mathbf{d}, \mathbf{x} \geq \mathbf{0}\} && \neq \emptyset \\
 \mathcal{X} &= \mathcal{D} \cap \mathbb{Z}^n
 \end{aligned} \tag{10}$$

# Feasible domain $\mathcal{A} \cap \mathcal{D}$ of the linear relaxation

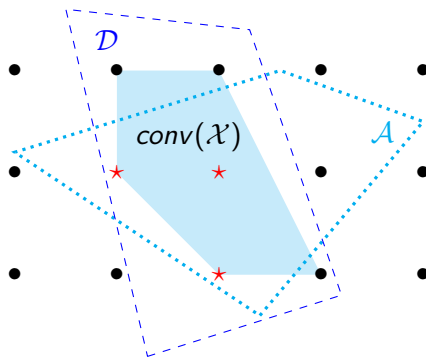


# Minkowski-Weyl theorem

Let  $\{\mathbf{x}_p\}_{p \in \Omega}$  and  $\{\mathbf{x}_r\}_{r \in \Gamma}$  be the sets of **extreme points** and **extreme rays** of  $\text{conv}(\mathcal{X})$ . **Any solution**  $x \in \text{conv}(\mathcal{X})$  can be written as follows:

$$\begin{aligned}\mathbf{x} &= \sum_{p \in \Omega} \mathbf{x}_p \theta_p + \sum_{r \in \Gamma} \mathbf{x}_r \theta_r \\ \sum_{p \in \Omega} \theta_p &= 1 \\ \theta_p &\geq 0 & \forall p \in \Omega \\ \theta_r &\geq 0 & \forall r \in \Gamma\end{aligned}\tag{11}$$

# Convex hull of $\mathcal{X}$ (only extreme points)



# D-W decomposition principle

Substitute (11) in (9) to yield the **integer master problem** (IMP), also called the **extended formulation**

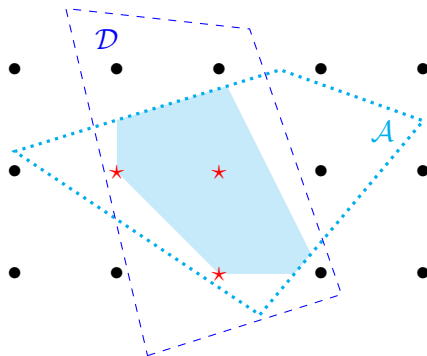
$$\begin{aligned}
 \min \quad & \sum_{p \in \Omega} c_p \theta_p \quad + \quad \sum_{r \in \Gamma} c_r \theta_r \\
 \text{s.t.} \quad & \sum_{p \in \Omega} \mathbf{a}_p \theta_p \quad + \quad \sum_{r \in \Gamma} \mathbf{a}_r \theta_r \geq \mathbf{b} \\
 & \sum_{p \in \Omega} \theta_p = 1
 \end{aligned} \tag{12}$$

$$\begin{aligned}
 \theta_p &\geq 0, \forall p \in \Omega, & \theta_r &\geq 0, \forall r \in \Gamma \\
 \sum_{p \in \Omega} \mathbf{x}_p \theta_p \quad + \quad & \sum_{r \in \Gamma} \mathbf{x}_r \theta_r = \mathbf{x} \in \mathbb{Z}^n
 \end{aligned}$$

where

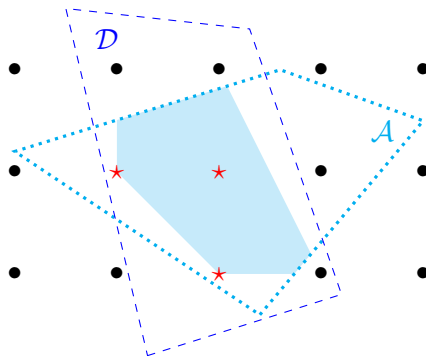
$$\begin{aligned}
 c_p &= \mathbf{c}^T \mathbf{x}_p, & \mathbf{a}_p &= \mathbf{A} \mathbf{x}_p, & \forall p \in \Omega \\
 c_r &= \mathbf{c}^T \mathbf{x}_r, & \mathbf{a}_r &= \mathbf{A} \mathbf{x}_r, & \forall r \in \Gamma
 \end{aligned} \tag{13}$$

# Domain of linear relaxation of IMP



IMP linear relaxation can yield a better lower bound than the linear relaxation of the compact formulation if the extreme points of  $\mathcal{D}$  are not all integer

# Domain of linear relaxation of IMP



IMP linear relaxation can yield a better lower bound than the linear relaxation of the compact formulation if the extreme points of  $\mathcal{D}$  are not all integer

# Path-flow formulation of the VRPTW

## Notation

$\Omega$ : set of feasible paths in  $G$  ( $\equiv$  extreme points of  $\text{conv}(\mathcal{X})$ )

$c_p$ : cost of feasible path  $p \in \Omega$

$v_{pi}$ : binary parameter equal to 1 if path  $p \in \Omega$  visits customer  $i$

$\theta_p$ : binary variable equal to 1 if path  $p \in \Omega$  is selected

$$\min \sum_{p \in \Omega} c_p \theta_p \quad (14) \quad \text{total cost}$$

$$\text{s.t.} \quad \sum_{p \in \Omega} v_{pi} \theta_p = 1, \quad \forall i \in N \quad (15) \quad \text{visit each customer}$$

$$\theta_p \in \{0, 1\}, \quad \forall p \in \Omega \quad (16) \quad \text{binary requirements}$$

# Path-flow formulation of the VRPTW

## Notation

$\Omega$ : set of feasible paths in  $G$  ( $\equiv$  extreme points of  $\text{conv}(\mathcal{X})$ )

$c_p$ : cost of feasible path  $p \in \Omega$

$v_{pi}$ : binary parameter equal to 1 if path  $p \in \Omega$  visits customer  $i$

$\theta_p$ : binary variable equal to 1 if path  $p \in \Omega$  is selected

$$\min \sum_{p \in \Omega} c_p \theta_p \quad (14) \quad \text{total cost}$$

$$\text{s.t.} \quad \sum_{p \in \Omega} v_{pi} \theta_p = 1, \quad \forall i \in N \quad (15) \quad \text{visit each customer}$$

$$\theta_p \in \{0, 1\}, \quad \forall p \in \Omega \quad (16) \quad \text{binary requirements}$$

## Remarks

- In practice, this model contains a **huge number of variables**
- For certain vehicle routing problems, **integrality requirements are not** on the route variables  $\theta_p$
- Often yields **better lower bounds** than arc-flow model
- Has **less symmetry** than arc-flow model with three-index arc-flow variables

## Remarks

- In practice, this model contains a **huge number of variables**
- For certain vehicle routing problems, **integrality requirements are not** on the route variables  $\theta_p$
- Often yields **better lower bounds** than arc-flow model
- Has **less symmetry** than arc-flow model with three-index arc-flow variables

# Outline

- 1 VRPTW definition
- 2 Mathematical formulations
  - Arc-flow formulation
  - Dantzig-Wolfe decomposition
  - Path-flow formulation
- 3 Branch-price-and-cut
  - Master problem
  - Pricing problem
  - Linear relaxation computational results
  - Branching
  - Cutting
  - Using upper bounds
- 4 Computational results (VRPTW and CVRP)

# Branch-price-and-cut algorithm

**Branch-and-bound** algorithm where lower bounds are computed by **column generation (CG)** and **cuts** are added to strengthen the linear relaxations

## Column generation

Iterative algorithm which alternates between solving a **restricted master problem (RMP)** and one or several **pricing problems (PP)**

- RMP corresponds to the linear relaxation of (14)–(16) restricted to a subset of variables
- In general, PP is an elementary shortest path problem with resource constraints (ESPPRC)

**Stop** when no more negative reduced cost routes exist

# Branch-price-and-cut algorithm

**Branch-and-bound** algorithm where lower bounds are computed by **column generation (CG)** and **cuts** are added to strengthen the linear relaxations

## Column generation

Iterative algorithm which alternates between solving a **restricted master problem (RMP)** and one or several **pricing problems (PP)**

- RMP corresponds to the linear relaxation of (14)–(16) restricted to a subset of variables
- In general, PP is an elementary shortest path problem with resource constraints (ESPPRC)

**Stop** when no more negative reduced cost routes exist

# Master problem and RMP

- The linear relaxation of the IMP (14)–(16) is called the **master problem (MP)**
- MP is solved by CG
- At a given CG iteration (with  $\hat{\Omega} \subset \Omega$ ), the **RMP** is

$$\min \sum_{p \in \hat{\Omega}} c_p \theta_p \quad (17)$$

$$\text{s.t.} \quad \sum_{p \in \hat{\Omega}} v_{pi} \theta_p = 1, \quad \forall i \in N \quad (\pi_i) \quad (18)$$

$$\theta_p \geq 0, \quad \forall p \in \hat{\Omega} \quad (19)$$

# Master problem and RMP

- The linear relaxation of the IMP (14)–(16) is called the **master problem (MP)**
- MP is solved by CG
- At a given CG iteration (with  $\hat{\Omega} \subset \Omega$ ), the **RMP** is

$$\min \sum_{p \in \hat{\Omega}} c_p \theta_p \quad (17)$$

$$\text{s.t.} \quad \sum_{p \in \hat{\Omega}} v_{pi} \theta_p = 1, \quad \forall i \in N \quad (\pi_i) \quad (18)$$

$$\theta_p \geq 0, \quad \forall p \in \hat{\Omega} \quad (19)$$

# Pricing problem

## Dual variables and reduced cost

$\pi_i, i \in N$ : dual variables associated with constraints (18)

$\pi_o = 0$ : defined for notational conciseness

**Reduced cost**  $\bar{c}_p$  of variable  $\theta_p$ :

$$\bar{c}_p = c_p - \sum_{i \in N} v_{pi} \pi_i = \sum_{(i,j) \in p} (c_{ij} - \pi_i) = \sum_{(i,j) \in p} \bar{c}_{ij}$$

$\bar{c}_{ij} = c_{ij} - \pi_i$ : **modified arc cost**

## PP

- Can be defined as:

Find one or several **negative reduced cost routes**  $p \in \Omega$  if at least one exists

- But **often expressed** as:

Find a route with the **least reduced cost**

$$\min_{p \in \Omega} \bar{c}_p$$

- Corresponds to an **ESPPRC** with modified arc costs  $\bar{c}_{ij}$ ,  $(i, j) \in A$

# ESPPRC

## ESPPRC definition

- Shortest path problem from  $o$  to  $d$
- Modified arc cost  $\bar{c}_{ij}$  to compute path reduced costs
- Elementarity requirements: Each customer must be visited at most once
- Resource constraints:
  - In general, a resource restricts path feasibility: time and load
  - It is a quantity that varies along a path and its value must fall within a prespecified resource window at each node
- Strongly NP-hard
- A relaxation such as the following (non-elementary) shortest path problem with resource constraints (SPPRC) can be used

# ESPPRC

## ESPPRC definition

- Shortest path problem from  $o$  to  $d$
- Modified arc cost  $\bar{c}_{ij}$  to compute path reduced costs
- Elementarity requirements: Each customer must be visited at most once
- Resource constraints:
  - In general, a resource restricts path feasibility: time and load
  - It is a quantity that varies along a path and its value must fall within a prespecified resource window at each node
- Strongly NP-hard
- A relaxation such as the following (non-elementary) shortest path problem with resource constraints (SPPRC) can be used

# SPPRC

- Same definition as ESPPRC except that **cycles are allowed** (example: path  $p = o - 1 - 2 - 3 - 1 - 5 - 6 - 3 - d$  contains cycles  $1 - 2 - 3 - 1$  and  $3 - 1 - 5 - 6 - 3$ )
- **Columns with  $v_{pi} \geq 2$**  can be generated
- Enlarged set of columns  $\Omega$  to consider in model (14)–(16)
- Model (14)–(16) remains valid because set partitioning constraints (15) and binary requirements (16) forbid columns with  $v_{pi} \geq 2$  to be part of an integer solution
- Yields a **weaker lower bound**
  - For the path  $p$  above,  $\theta_p = 0.5$  completely covers customers 1 and 3

# Labeling algorithm

- SPPRC is usually solved by dynamic programming, more precisely, by a **labeling algorithm**
- In such an algorithm, partial paths start at the source node  $o$  and are represented by multi-dimensional resource vectors, called **labels**
- Starting from an initial label associated with node  $o$ , labels are propagated forwardly using **resource extension functions** (REFs) through network  $G$  (backward labeling is also possible)
- **Resource windows** are checked at each node to discard infeasible partial paths
- To avoid enumerating all feasible paths, a **dominance rule** is applied to discard unpromising labels
- **Multiple labels** are associated with each node

# Labels for the VRPTW

- A **label**  $E_i^p$  representing a partial path  $p$  from node  $o$  to a node  $i$  contains three components
  - $Z_i^p$ : (reduced) **cost** of  $p$
  - $L_i^p$ : **load** accumulated along  $p$
  - $T_i^p$ : (earliest) **start of service time** at  $i$
- $E_i^p = (Z_i^p, L_i^p, T_i^p)$  is **feasible** if

$$L_i^p \in [0, Q] \quad \text{and} \quad T_i^p \in [a_i, b_i]$$

- No restrictions are imposed on the cost component  $Z$  which is **also viewed** as an unrestricted resource
- Whenever unambiguous, indices  $p$  and  $i$  may be omitted in the following

# Resource extension functions

- Path  $p$  (label  $E_i$ ) **can be extended** by appending arc  $(i, j) \in A$
- Resulting path is represented by a label  $E_j = (Z_j, L_j, T_j)$  whose components are computed using the following **REFs**

$$Z_j = f_{ij}^{cost}(E_i) = Z_i + \bar{c}_{ij}$$

$$L_j = f_{ij}^{load}(E_i) = L_i + q_j$$

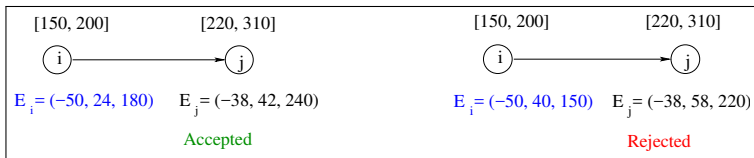
$$T_j = f_{ij}^{time}(E_i) = \max \{a_j, T_i + t_{ij}\}$$

- REFs  $f_{ij}^{time}()$  ensure that time window lower bounds are always respected
- Therefore,  $E_j$  is feasible if  $L_j \leq Q$  and  $T_j \leq b_j$

# Label extension: example

Consider

- a label  $E_i = (Z_i, L_i, T_i)$  for a path ending in node  $i$
- $[a_i, b_i] = [150, 200]$ ,  $[a_j, b_j] = [220, 310]$
- $\bar{c}_{ij} = 12$ ,  $t_{ij} = 60$ ,  $q_j = 18$ ,  $Q = 50$



# Dominance

## Principle

- Consider two labels ( $E^1$  and  $E^2$ ) representing two feasible partial paths ending at the same node
- $E^2$  cannot yield a shortest  $o-d$  path if
  - Every feasible extension of  $E^2$  is also a feasible extension of  $E^1$
  - For every such extension, the cost of the path resulting from the extension of  $E^1$  is less than or equal to the cost of the path resulting from the extension of  $E^2$
- In this case, we say that label  $E^1$  dominates label  $E^2$ , which can then be discarded
- In general, these conditions cannot be verified easily

When all **REFs are non-decreasing**, the following (more restrictive) dominance rule can be applied

### Dominance rule

Label  $E^1$  **dominates** label  $E^2$  if  $E^1 \leq E^2$  componentwise (that is,  $Z^1 \leq Z^2$ ,  $L^1 \leq L^2$  and  $T^1 \leq T^2$ ). In this case,  $E^2$  can be **discarded**.

When all components are equal, keep one of the two labels

### Example

- Consider three labels  
 $E^1 = (-32, 25, 200)$ ,  $E^2 = (-27, 34, 230)$ ,  
 $E^3 = (-10, 20, 180)$
- $E^1$  dominates  $E^2$
- $E^1$  does not dominate  $E^3$  and vice versa (keep both labels)

# Algorithm

## Notation

$\mathcal{U}_i$ : set of unprocessed labels at node  $i$

$\mathcal{P}_i$ : set of processed labels at node  $i$

$i(E)$ : last node of the path associated with  $E$

$DOM(\mathcal{U}_j, \mathcal{P}_j)$ : dominance algorithm applied to labels in  $\mathcal{U}_j$  and  $\mathcal{P}_j$   
that returns a possibly reduced set  $\mathcal{U}_j$

---

**Algorithm 1** : Generic SPPRC labeling algorithm
 

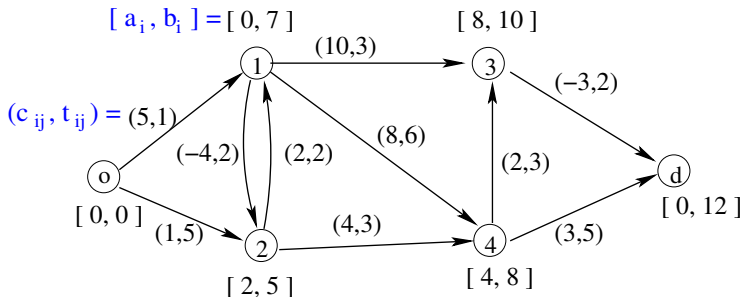
---

- 1: Set  $\mathcal{U}_i = \mathcal{P}_i = \emptyset$  for all  $i \in \bar{N}$  except  $\mathcal{U}_o = \{(0, 0, a_o)\}$
  - 2: **while**  $\bigcup_{i \in \bar{N}} \mathcal{U}_i \neq \emptyset$  **do**
  - 3:   Choose a label  $E \in \bigcup_{i \in \bar{N}} \mathcal{U}_i$  and remove  $E$  from  $\mathcal{U}_{i(E)}$
  - 4:   **for all** arcs  $(i(E), j) \in A$  **do**
  - 5:     Extend  $E$  along  $(i(E), j)$  using the REFs to create a new label  $E'$
  - 6:     **if**  $E'$  is feasible **then**
  - 7:       Add  $E'$  to  $\mathcal{U}_j$
  - 8:        $\mathcal{U}_j = \text{DOM}(\mathcal{U}_j, \mathcal{P}_j)$
  - 9:   Add  $E$  to  $\mathcal{P}_{i(E)}$
  - 10: Filter  $\mathcal{P}_d$  to find a shortest  $o$ - $d$  path
-

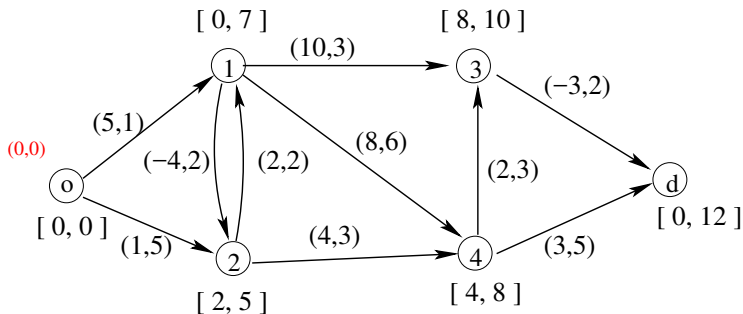
## Remarks

- If there exists a resource with a positive consumption on every arc (e.g., time), then label  $E$  can be chosen in Step 3 as the one with the minimal value for this resource
  - Yields a **pseudo-polynomial** algorithm for the VRPTW
  - State is given by node, load and time values
  - Dimension of state space =  $O(|N| \times Q \times W)$   
where  $W$  is the maximum width of a resource window
  - Using this label selection rule, a state cannot be visited more than once
- In Step 8, dominance can be applied only when needed
- A **pulling algorithm** (labels are extended along arcs  $(i, j)$  for a fixed  $j$ ) might be more efficient because it allows to apply dominance on a subset of new labels associated with the same node

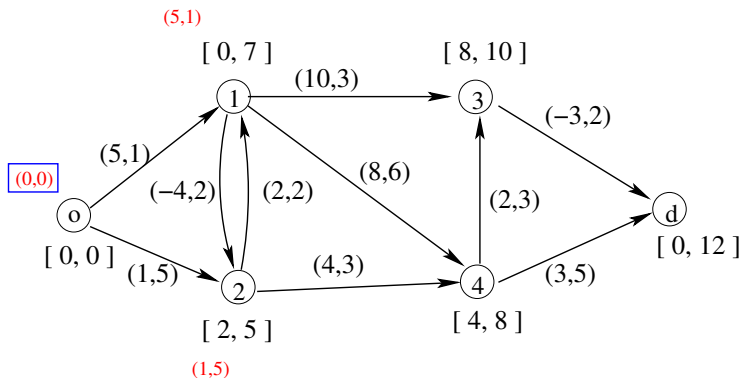
# Example without capacity constraint



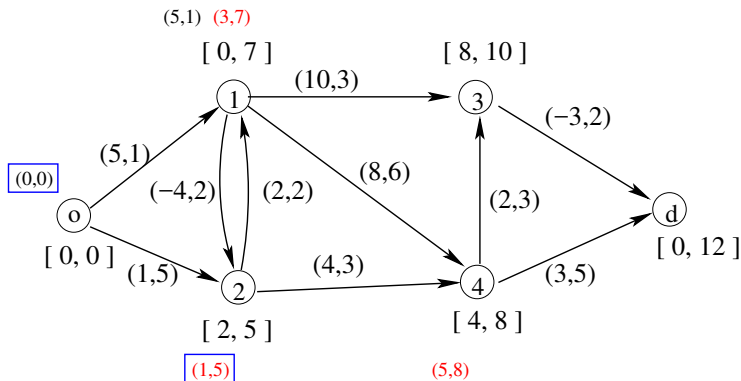
## Initial label



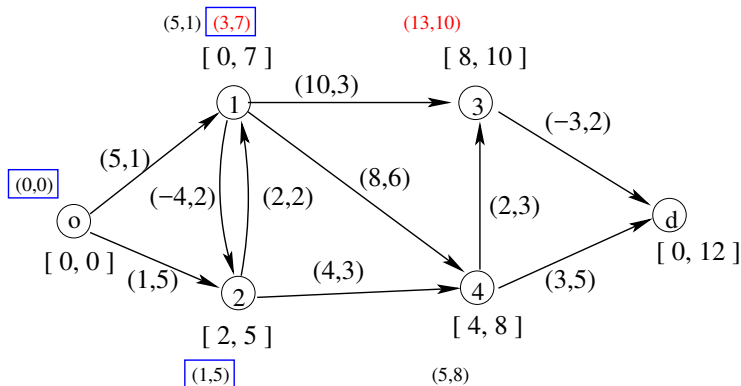
# Extension of label $(0,0)$



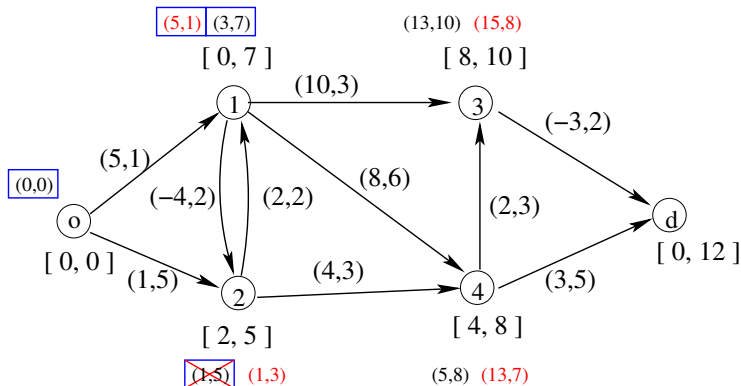
# Extension of label (1, 5) (least cost)



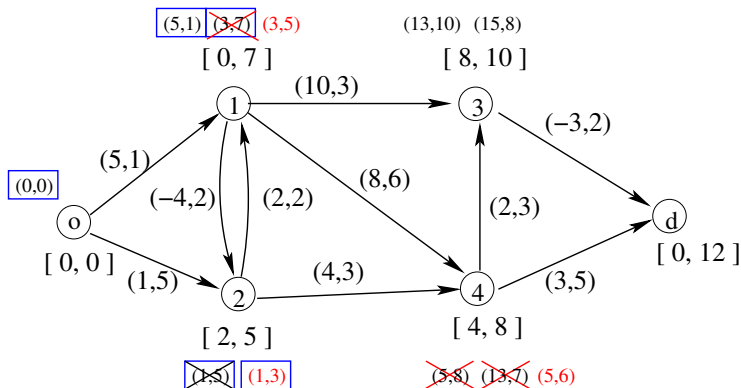
# Extension of label (3,7)



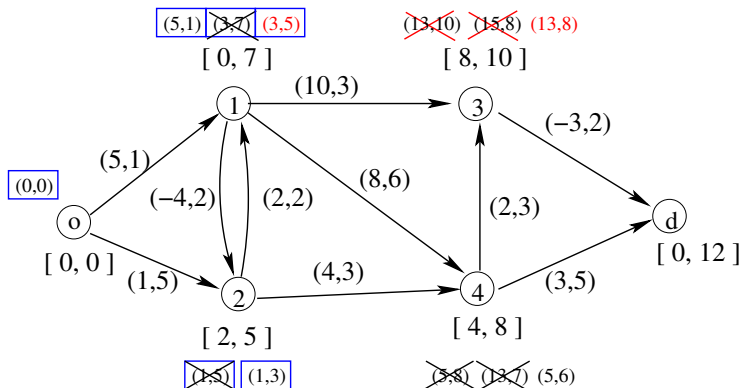
# Extension of label (5, 1)



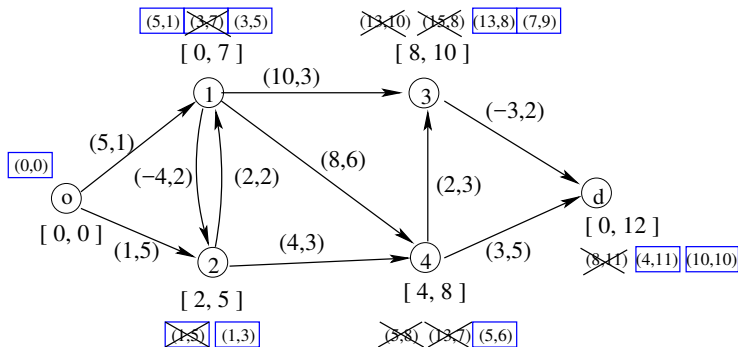
# Extension of label (1,3)



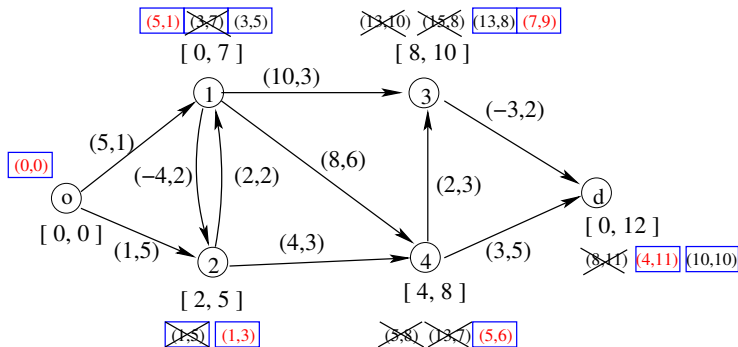
# Extension of label (3,5)



and so on



The shortest path is retrieved by finding the **preceding labels**



shortest path =  $o-1-2-4-3-d$

optimal cost = 4

optimal path earliest arrival time at node  $d$  = 11

## ESPPRC

## Basic version

- Also solved by a labeling algorithm
- **One additional resource per customer**  $k \in N$  that indicates whether this customer has been visited or not
- Corresponding **resource variables**:  $T^{cust_k}$ ,  $k \in N$
- **REFs** for these resources and arc  $(i, j)$ :

$$T_j^{cust_k} = \begin{cases} T_i^{cust_k} + 1 & \text{if } j = k \\ T_i^{cust_k} & \text{otherwise} \end{cases} \quad \forall k \in N$$

- Yields labels with  **$3 + |N|$  dimensions**
- New label is **feasible** if  $T_j^{cust_k} \leq 1$ ,  $\forall k \in N$
- **Additional dominance conditions**:  $T^{1, cust_k} \leq T^{2, cust_k}$ ,  $\forall k \in N$
- No pseudo-polynomial time algorithms

## Improved version

- The additional resources indicate whether the customers are **reachable or not**
- A customer  $k$  is **unreachable from a label  $E_i$**  at node  $i$  (that is,  $U_k(L_i, T_i) = 1$ ) if at least one of the following three conditions holds (assuming triangle inequality for travel times)
  - Customer  $k$  has already been visited
  - Load prevents reaching  $k$  ( $L_i + q_k > Q$ )
  - Time prevents reaching  $k$  ( $T_i + t_{ik} > b_k$ )

- **REFs become:**

$$T_j^{cust_k} = \begin{cases} T_i^{cust_k} + 1 & \text{if } j = k \\ \max \{ T_i^{cust_k}, U_k(L_j, T_j) \} & \text{otherwise} \end{cases} \quad \forall k \in N$$

- Allows **more dominance**

## Example

- Consider two paths  $p_1 = o - 1 - 3$  and  $p_2 = o - 2 - 3$
- For the basic version,  $p_1$  and  $p_2$  yield labels  $E^1$  and  $E^2$

label	$Z$	$L$	$T$	$T^{cust_1}$	$T^{cust_2}$	$T^{cust_3}$
$E^1$	-12	15	100	1	0	1
$E^2$	-8	25	110	0	1	1

- In this case,  $E^1$  does not dominate  $E^2$
- Assume  $[a_1, b_1] = [60, 115]$ ,  $[a_2, b_2] = [40, 125]$ ,  
 $t_{31} = 15$ ,  $t_{32} = 20$ ,  $q_1 = 5$ ,  $q_2 = 15$ ,  $Q = 80$
- For the improved version,  $p_1$  and  $p_2$  yield labels  $E^1$  and  $E^2$

label	$Z$	$L$	$T$	$T^{cust_1}$	$T^{cust_2}$	$T^{cust_3}$
$E^1$	-12	15	100	1	0	1
$E^2$	-8	25	110	1	1	1

- In this case,  $E^1$  dominates  $E^2$

# Bidirectional search

- Select a bounded resource (e.g., time)
- Select a middle value  $M$  for this resource ( $M = (b_d - a_o)/2$ )
- From the source node  $o$ , generate **forward labels**
  - Forward extension of a label  $E_i$  along arc  $(i, j)$  is allowed if  $T_i \leq M$
- From the sink node  $d$ , generate **backward labels** using backward REFs
  - Backward extension of a label  $E_j$  along arc  $(i, j)$  is allowed if  $\min \{b_i, T_j - t_{ij}\} > M$
- At each node  $i \in \tilde{N}$ , **join forward and backward labels** to identify feasible  $o$ - $d$  paths
- Among these paths, find the shortest one

## Remark

Midpoint can be determined dynamically to balance the number of forward and backward labels

# Bidirectional search

- Select a bounded resource (e.g., time)
- Select a middle value  $M$  for this resource ( $M = (b_d - a_o)/2$ )
- From the source node  $o$ , generate **forward labels**
  - Forward extension of a label  $E_i$  along arc  $(i, j)$  is allowed if  $T_i \leq M$
- From the sink node  $d$ , generate **backward labels** using backward REFs
  - Backward extension of a label  $E_j$  along arc  $(i, j)$  is allowed if  $\min \{b_i, T_j - t_{ij}\} > M$
- At each node  $i \in \tilde{N}$ , **join forward and backward labels** to identify feasible  $o$ - $d$  paths
- Among these paths, find the shortest one

## Remark

Midpoint can be determined dynamically to balance the number of forward and backward labels

# Decremental state-space relaxation

Add customer resources **gradually, as needed**

- ① Start without any customer resources
- ② Solve the problem with the current set of customer resources
- ③ If the computed optimal path contains a cycle
  - ① Add a customer resource to forbid this cycle
  - ② Return to Step 2

## Remarks

- In Step 3.1, there are several ways to select the resource
- In a column generation context
  - Add a condition to the test in Step 3: the current optimal value is negative
  - In Step 1, start with the set of customer resources used at the end of the previous column generation iteration
  - Stop at Step 3 if at least one negative reduced cost elementary path is found

# Decremental state-space relaxation

Add customer resources **gradually, as needed**

- ① Start without any customer resources
- ② Solve the problem with the current set of customer resources
- ③ If the computed optimal path contains a cycle
  - ① Add a customer resource to forbid this cycle
  - ② Return to Step 2

## Remarks

- In Step 3.1, there are several ways to select the resource
- In a column generation context
  - Add a condition to the test in Step 3: the current optimal value is negative
  - In Step 1, start with the set of customer resources used at the end of the previous column generation iteration
  - Stop at Step 3 if at least one negative reduced cost elementary path is found

# Other relaxations

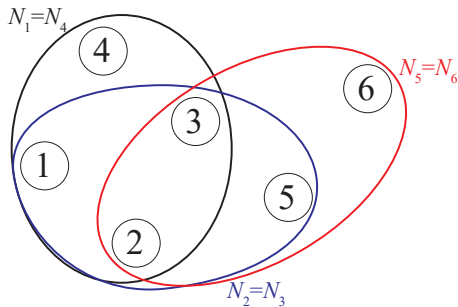
## SPPRC with $k$ -cycle elimination

- In a path, a  **$k$ -cycle** occurs when the same node is visited twice with  $k - 1$  nodes between these two visits
- Example:  $p = o - 1 - 2 - 3 - 1 - 4 - 5 - 4 - d$  contains a 2-cycle (for node 4) and a 3-cycle (for node 1)
- **SPPRC with  $k$ -cycle elimination** forbids paths with  $l$ -cycles for  $l = 2, \dots, k$
- Yields **stronger lower bounds** than SPPRC
- **Easy to implement for  $k = 2$**
- Can be used in decremental state-space relaxation for ESPPRC

### SPPRC with neighborhoods (*ng*-paths)

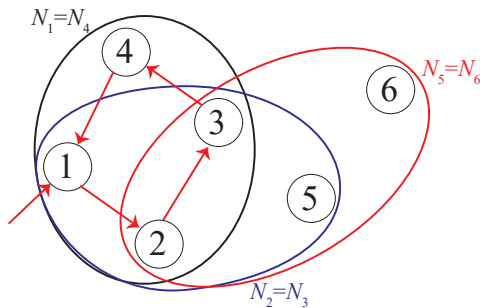
- Associate with each node  $i \in N$  a **neighborhood**  $N_i \subset N$  (e.g., node  $i$  and its closest nodes)
- **Allows a cycle** for a node  $i$  only if this cycle contains a node  $j$  for which  $i \notin N_j$  (returning to  $i$  from  $j$  is a detour)

# Examples



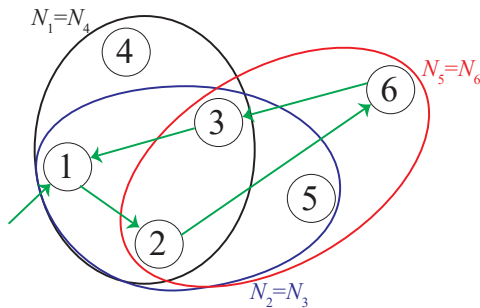
Node neighborhoods

# Examples



Infeasible cycle

# Examples



Feasible cycle

## Treatment in the labeling algorithm

- Use **customer resources**  $T^{cust_k}$ ,  $k \in N$
- For a path  $p = o - i_1 - \dots - i_h$ ,  $T^{cust_k}$  is equal to 1 if  $k$  belongs to  $p$  (that is,  $\exists g \in \{1, \dots, h\}$  such that  $k = i_g$ ) and  $k \in N_{i_\ell}$  for all  $\ell \in \{g, \dots, h\}$  and 0 otherwise
- Thus,  $T^{cust_k} = 0$  for a node  $k$  if  $k$  has not been visited or if it does not belong to the neighborhood of a node visited after its last visit
- **REFs** for arc  $(i, j)$

$$T_j^{cust_k} = \begin{cases} 1 & \text{if } j = k \text{ or } (T_i^{cust_k} = 1 \text{ and } k \in N_j) \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in N$$

- At node  $i$ , **number of customer resources at 1** is less than or equal to  $|N_i|$

## Neighborhood size

- If  $N_i = N$  for all  $i \in N$ , then only elementary routes
- If  $N_i = \{i\}$  for all  $i \in N$ , then all cycles are allowed
- In practice,  $10 \leq |N_i| \leq 15$  for all  $i \in N$  offers a good compromise

## Dynamic neighborhoods

- Start with small neighborhoods
- Add neighbors to some neighborhoods
  - At each column generation iteration
  - Only once the master problem is solved

## Neighborhood size

- If  $N_i = N$  for all  $i \in N$ , then only elementary routes
- If  $N_i = \{i\}$  for all  $i \in N$ , then all cycles are allowed
- In practice,  $10 \leq |N_i| \leq 15$  for all  $i \in N$  offers a good compromise

## Dynamic neighborhoods

- Start with small neighborhoods
- Add neighbors to some neighborhoods
  - At each column generation iteration
  - Only once the master problem is solved

# Heuristics

- Solving the ESPPRC (or the *ng*-SPPRC with large neighborhoods) exactly might be **highly time consuming**
- Its exact solution is required only **for proving the optimality** of the current MP solution
- **Fast heuristics** can be used to generate negative reduced cost columns
  - Heuristic labeling algorithm
  - Local search (meta)heuristics

## Heuristic labeling algorithm

- Use a **reduced network**
  - Omit arcs based on their cost or their modified cost
- Use a **relaxed dominance rule**
  - Consider a subset of the customer resources
  - Increases the number of dominated labels
  - All customer resources are considered to check feasibility

## Tabu search heuristic

- Each column in the basis of the current MP solution is associated with a **zero reduced cost path**
- These paths are **good initial solutions** for a tabu search algorithm
- **Tabu search algorithm**
  - Iterative method that explores a neighborhood at each iteration to find the best neighbor solution (cost might deteriorate)
  - Neighborhood defined by **feasible moves**
    - **Insert or remove a node** from the current path
  - Selected move remains **tabu** (cannot be undone) for a fixed number of iterations
  - **Stop** after a fixed number of iterations (switch to next initial solution)

# Column generator usage

## Column generator characteristics

- **Tabu search**: very fast (when small number of iterations per initial column), generates a large number of columns in the early CG iterations, efficiency decreases rapidly toward the end
- **Heuristic labeling**: fast, more efficient than tabu search to find columns but generates less columns in the early CG iterations
- **Exact labeling**: very slow, highly efficient to find columns

## Usage

At each CG iteration

- Use tabu search algorithm
- **If it fails**, use heuristic labeling algorithm
- **If it fails again**, use exact labeling algorithm

# Solomon's VRPTW instances

- 56 instances with 100 customers
- 6 classes: R1, C1, RC1, R2, C2, RC2
  - R: random customer locations
  - C: clustered customer locations
  - RC: mixed locations
  - 1: relatively tight time windows
  - 2: large time windows

# Linear relaxation results (with or without tabu)

	with tabu					without tabu			
Instance	cpu (s)	nb it	nb tabu	nb hdp	nb edp	cpu (s)	nb it	nb hdp	nb edp
RC104	36	77	57	19	1	35	152	149	3
RC108	26	56	40	14	2	42	136	131	5
R108	26	84	66	17	1	23	102	99	3
R112	30	61	47	12	2	23	61	59	2
RC203	95	117	103	13	1	1131	608	599	9
RC206	68	140	116	23	1	78	449	448	1
RC207	148	133	104	28	1	397	374	371	3
R203	172	214	174	39	1	758	613	601	12
R205	97	170	135	34	1	77	351	349	2
R206	210	183	137	45	1	864	511	503	8
R209	181	204	164	39	1	278	405	403	2
R210	269	184	141	41	2	1420	451	438	13

From Desaulniers et al. (2008)

# Lower bounds (easy RC and R instances)

Instance	spprc	spprc 2-ce	spprc 3-ce	espprc	UB
RC101	1567.5	1584.1	<i>id</i>	<i>id</i>	1619.8
RC102	1380.2	1403.6	1404.9	1406.3	1457.4
RC103	1170.3	1218.5	1221.7	1225.5	1258.0
RC104	1052.6	1094.3	1097.2	1101.8	1132.3
RC105	1453.9	1471.2	1471.8	1471.9	1513.7
RC106	1249.0	1308.8	1317.2	1318.8	1372.7
RC107	1117.3	1170.7	1181.1	1183.4	1207.8
RC108	1035.9	1063.0	1066.8	1073.4	1114.2
R101	1631.1	<i>id</i>	<i>id</i>	<i>id</i>	1637.7
R102	1466.6	<i>id</i>	<i>id</i>	<i>id</i>	1466.6
R103	1203.2	1206.3	1206.5	1206.8	1208.7
R104	937.1	949.1	955.3	956.9	971.5
R105	1341.2	1346.1	<i>id</i>	<i>id</i>	1355.3
R106	1212.3	1226.4	1226.4	1226.9	1234.6
R107	1037.0	1051.8	1052.2	1053.2	1064.6
R108	891.7	907.2	911.5	913.5	932.1
R109	1097.5	1130.6	1134.3	<i>id</i>	1146.9
R110	1021.3	1048.5	1052.5	1055.6	1068.0
R111	1005.9	1032.0	1034.2	1034.7	1048.7
R112	892.5	919.2	923.6	926.7	948.6

From Desaulniers et al. (2008)

# Lower bounds (difficult RC and R instances)

Instance	spprc	spprc 2-ce	spprc 3-ce	espprc	UB
RC201	1108.9	1240.4	1255.4	1255.9	1261.8
RC202	882.6	1004.1	1050.9	1088.1	1092.3
RC203	698.1	815.3	-	922.5	923.7
RC204	608.6	688.3	-	-	783.5
RC205	967.1	1055.5	1117.9	1147.6	1154.0
RC206	852.2	952.2	1005.9	1038.6	1051.1
RC207	769.1	866.3	886.9	947.3	962.9
RC208	627.2	704.6	-	-	776.1
R201	1080.7	1136.2	1140.0	1140.3	1143.2
R202	934.2	1009.8	-	1022.2	1029.6
R203	756.7	846.5	-	866.9	870.8
R204	640.4	689.8	-	-	731.3
R205	838.4	916.6	930.6	938.9	949.8
R206	749.1	834.6	-	866.9	875.9
R207	668.9	747.6	-	790.7	794.0
R208	610.7	661.7	-	-	701.0
R209	750.4	819.8	833.9	841.4	854.8
R210	754.0	849.3	-	889.4	900.5
R211	650.8	705.8	-	-	746.7

From Desaulniers et al. (2008)

# Branching on the MP variables

- In a typical branch-and-bound algorithm, we can branch directly on the model variables
- Here, this would correspond to imposing the decisions:

$$\theta_p = 0 \quad \text{and} \quad \theta_p = 1$$

- Usually inefficient
  - Branch  $\theta_p = 1$  is **strong** (good increase of LB)
  - Branch  $\theta_p = 0$  is **very weak** (close to no improvement of LB)
- Difficult implementation
  - Both decisions imposed in the MP
  - Decision  $\theta_p = 0$  **destroys the PP structure** because  $p$  must not be generated again

# Branching on the arc-flow formulation variables

## Principle

If you can define branching rules on the variables of the arc-flow formulation, then you can easily rewrite them in terms of the MP variables or impose them directly in the PP

In the following

$$X_{ij} = \sum_{k \in K} X_{ij}^k = \sum_{p \in \Omega} v_{ijp} \theta_p,$$

where  $v_{ijp} = 1$  if  $(i, j)$  is in path  $p$ , 0 otherwise

Different rules on

- 1 Number of vehicles used
- 2 Arc-flow variables

These are the most popular; other rules exist

# Branching on the arc-flow formulation variables

## Principle

If you can define branching rules on the variables of the arc-flow formulation, then you can easily rewrite them in terms of the MP variables or impose them directly in the PP

## In the following

$$X_{ij} = \sum_{k \in K} X_{ij}^k = \sum_{p \in \Omega} v_{ijp} \theta_p,$$

where  $v_{ijp} = 1$  if  $(i, j)$  is in path  $p$ , 0 otherwise

## Different rules on

- 1 Number of vehicles used
- 2 Arc-flow variables

These are the most popular; other rules exist

# Branching on the arc-flow formulation variables

## Principle

If you can define branching rules on the variables of the arc-flow formulation, then you can easily rewrite them in terms of the MP variables or impose them directly in the PP

## In the following

$$X_{ij} = \sum_{k \in K} X_{ij}^k = \sum_{p \in \Omega} v_{ijp} \theta_p,$$

where  $v_{ijp} = 1$  if  $(i, j)$  is in path  $p$ , 0 otherwise

## Different rules on

- 1 Number of vehicles used
- 2 Arc-flow variables

These are the most popular; other rules exist

# Number of vehicles used

- The **number of vehicles used** in a solution is  $V = \sum_{p \in \Omega} \theta_p$
- If  $V$  takes a **fractional** value  $\tilde{V}$ , then one can impose the rules

$$\sum_{p \in \Omega} \theta_p \leq \lfloor \tilde{V} \rfloor \quad \text{and} \quad \sum_{p \in \Omega} \theta_p \geq \lceil \tilde{V} \rceil$$

## Treatment

- These decisions are **global** (they cannot be verified one route at a time) and **imposed in the MP** by adding the corresponding constraints
- Let  $\mu$  be the dual variable associated with such a decision
- Because  $\sum_{p \in \Omega} \theta_p = \sum_{(o,j) \in A} X_{oj}$ , we see that  $\mu$  can be subtracted from the (modified) arc cost of every arc  $(o,j) \in A$

# Number of vehicles used

- The **number of vehicles used** in a solution is  $V = \sum_{p \in \Omega} \theta_p$
- If  $V$  takes a **fractional** value  $\tilde{V}$ , then one can impose the rules

$$\sum_{p \in \Omega} \theta_p \leq \lfloor \tilde{V} \rfloor \quad \text{and} \quad \sum_{p \in \Omega} \theta_p \geq \lceil \tilde{V} \rceil$$

## Treatment

- These decisions are **global** (they cannot be verified one route at a time) and **imposed in the MP** by adding the corresponding constraints
- Let  $\mu$  be the dual variable associated with such a decision
- Because  $\sum_{p \in \Omega} \theta_p = \sum_{(o,j) \in A} X_{oj}$ , we see that  $\mu$  can be subtracted from the (modified) arc cost of every arc  $(o,j) \in A$

# Arc-flow variables

- In a feasible solution, the flow  $X_{ij}$  on an arc  $(i,j)$  must take value 0 or 1 in an integer solution
- If  $X_{ij}$  takes a fractional value, then one can impose the rules

$$X_{ij} = 0 \quad \text{and} \quad X_{ij} = 1$$

## Treatment

- These decisions are local and imposed in the PP by modifying network  $G$ 
  - For  $X_{ij} = 0$ , remove arc  $(i,j)$
  - For  $X_{ij} = 1$ , remove all arcs  $(i,j')$  with  $j' \neq j$  and all arcs  $(i',j)$  with  $i' \neq i$
- The columns in the current RMP that do not respect the imposed decision must be removed from it

# Arc-flow variables

- In a feasible solution, the flow  $X_{ij}$  on an arc  $(i,j)$  must take value 0 or 1 in an integer solution
- If  $X_{ij}$  takes a fractional value, then one can impose the rules

$$X_{ij} = 0 \quad \text{and} \quad X_{ij} = 1$$

## Treatment

- These decisions are local and imposed in the PP by modifying network  $G$ 
  - For  $X_{ij} = 0$ , remove arc  $(i,j)$
  - For  $X_{ij} = 1$ , remove all arcs  $(i,j')$  with  $j' \neq j$  and all arcs  $(i',j)$  with  $i' \neq i$
- The columns in the current RMP that do not respect the imposed decision must be removed from it

# Cut types

## Robust cuts

- Can be expressed using **arc-flow variables**
- Duals can be transferred directly on the modified arc costs  $\bar{c}_{ij}$
- Structure of PP is not altered

## Non-robust cuts

- Defined directly on the **MP variables**  $\theta_p$
- Duals **cannot** be transferred directly on the modified arc costs  $\bar{c}_{ij}$
- Structure of PP is altered (typically, additional resources, modified dominance rule)

# Cut types

## Robust cuts

- Can be expressed using **arc-flow variables**
- Duals can be transferred directly on the modified arc costs  $\bar{c}_{ij}$
- Structure of PP is not altered

## Non-robust cuts

- Defined directly on the **MP variables**  $\theta_p$
- Duals **cannot** be transferred directly on the modified arc costs  $\bar{c}_{ij}$
- Structure of PP is altered (typically, additional resources, modified dominance rule)

# Robust $k$ -path inequalities

- Generalization of the capacity inequalities for the CVRP
- $S \subseteq N$ : subset of customers
- $\delta^-(S) \subset A$ : subset of arcs  $(i, j)$  entering  $S$  ( $i \notin S$  and  $j \in S$ )
- $k(S)$ : **lower bound** on the number of vehicles required to service the customers in  $S$
- **Corresponding  $k$ -path inequality**

$$\sum_{(i,j) \in \delta^-(S)} x_{ij} = \sum_{p \in \Omega} \sum_{(i,j) \in \delta^-(S)} v_{pij} \theta_p \geq k(S)$$

## Computation of $k(S)$

- For a given set  $S$ , lower bounds are given by
  - ①  $lb_1$  = Optimal value of a bin packing problem with items of length  $q_i$ ,  $i \in S$ , and bins of length  $Q$  (**NP-hard**)
  - ②  $lb_2 = \left\lceil \sum_{i \in S} q_i / Q \right\rceil$  (**easy to compute but weaker**)
  - ③  $lb_3$  = Optimal value of a vehicle scheduling problem that minimizes the number of vehicles required to service the customers in  $S$  considering only the time windows (**NP-hard**)
  - ④  $lb_4 = 2$  if the traveling salesman problem with time windows (TSPTW) for  $S$  is infeasible and 1 otherwise (**still NP-hard but easier to compute**)
- In practice,  $k(S)$  is set to  $\max \{lb_2, lb_4\}$
- When time windows are more constraining than capacity,  
 $lb_2 \leq lb_4$  and  $k(S) = lb_4 \Rightarrow$  **only 2-path cuts** can be found

## Example

- Consider three customers 1, 2 and 3 with  $[a_1, b_1] = [10, 15]$ ,  $[a_2, b_2] = [20, 25]$ ,  $[a_3, b_3] = [30, 40]$ ,  $t_{12} = 15$ ,  $t_{23} = 20$ ,  $t_{13} = 25$
- Arcs  $(1, 2)$ ,  $(2, 3)$  and  $(1, 3)$  exist
- To use arc  $(2, 3)$ , **start of service time at 2 must equal 20**
- Assume a fractional solution with three paths  $p_1$ ,  $p_2$  and  $p_3$  such that
  - $p_1$  contains arc  $(1, 2)$  and  $\theta_{p_1} = 0.5$
  - $p_2$  contains arc  $(2, 3)$  and  $\theta_{p_2} = 0.5$
  - $p_3$  contains arc  $(1, 3)$  and  $\theta_{p_3} = 0.5$
- For  $S = \{1, 2, 3\}$ ,  $k(S) = 2$  and the corresponding 2-path inequality is violated by this solution

## Separation

- No efficient algorithm yet
- Enumerate some sets  $S$
- For each set
  - Compute  $lb_2$
  - If  $\sum_{(i,j) \in \delta^-(S)} X_{ij} < lb_2$ , then a cut is found with  $k(S) = lb_2$
  - Else if  $1 < \sum_{(i,j) \in \delta^-(S)} X_{ij} < 2$ , then solve a TSPTW to compute  $lb_4$ 
    - If  $lb_4 = 2$ , then a cut is found with  $k(S) = lb_4$

## Treatment

- Add the following cut to the MP

$$\sum_{p \in \Omega} \sum_{(i,j) \in \delta^-(S)} v_{pij} \theta_p \geq k(S)$$

- In the PP, its dual variable is subtracted from the modified cost of all arcs in  $\delta^-(S)$
- **Weak** version of the cuts because a same path  $p$  contributes more than once if it enters  $S$  more than once  
(  $\sum_{(i,j) \in \delta^-(S)} v_{pij} > 1$  )

# Non-robust subset row inequalities

- Defined directly on the **MP variables**  $\theta_p$
- Chvátal-Gomory inequalities of rank 1
- $S$ : subset of customers (rows)

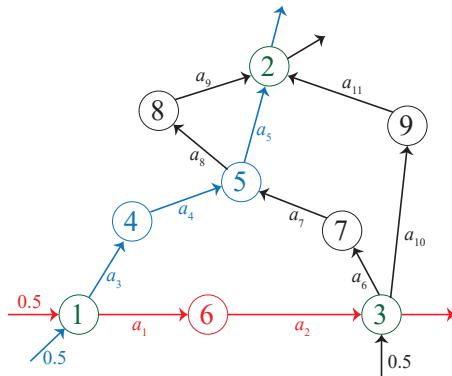
$$\sum_{p \in \Omega} \left\lfloor \frac{1}{k} \sum_{i \in S} v_{pi} \right\rfloor \theta_p \leq \left\lfloor \frac{|S|}{k} \right\rfloor, \quad \forall S \subseteq N, 2 \leq k < |S|$$

- Applied for **subsets of three customers** ( $|S| = 3, k = 2$ )  
 $\Rightarrow$  clique cuts defined for three customers

$$\sum_{p \in \Omega} l_{pS} \theta_p \leq 1, \quad \forall S \subseteq N, |S| = 3$$

where  $l_{pS} = 1$  if path  $p$  visits at least two customers in  $S$

# Example



$$S = \{1, 2, 3\} \text{ and } \sum_{p \in \Omega} I_{pS} \theta_p = 0.5 + 0.5 + 0.5 = 1.5 \not\leq 1$$

## Separation

- NP-hard separation problem
- For  $|S| = 3$ , **complete enumeration** of all subsets  $S$

## Treatment

- $\mathcal{SR}$ : set of subsets  $S$  with a subset row cut
- Add the cuts for  $S \in \mathcal{SR}$  in MP (non-positive dual variables  $\sigma^S$ )
- In the PP
  - Add **one unrestricted resource for each cut**
    - $R^S$  = number of customers visited in subset  $S$  **modulo 2**
  - **When extending a label** along an arc  $(i, j)$ , subtract  $\sigma^S$  from  $Z_j$  if  $R_i^S = 1$  and  $R_j^S = 0$  for all  $S \in \mathcal{SR}$
  - In the **dominance rule**

$$Z_1 - \sum_{S \in \mathcal{SR} : R_1^S=1 \ \& \ R_2^S=0} \sigma^S \leq Z_2$$

## Remarks

- Other versions with 1, 4, and 5 rows and with different C-G multipliers
- Very efficient at closing integrality gap
- PP is much more difficult to solve

# Limited-memory SRCs

- Goal
  - Reduce the difficulty of solving the PP
- How ?
  - **Increase dominance** by resetting to 0 the number of customers covered in  $S$  if the route leaves the memory (**weaker lifting**)
- Drawback
  - **Weakened cuts**

## Two memory types

- Node memory
- Arc memory

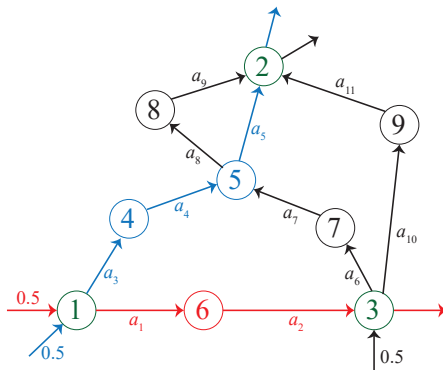
# Limited-memory SRCs

- Goal
  - Reduce the difficulty of solving the PP
- How ?
  - **Increase dominance** by resetting to 0 the number of customers covered in  $S$  if the route leaves the memory (**weaker lifting**)
- Drawback
  - **Weakened cuts**

## Two memory types

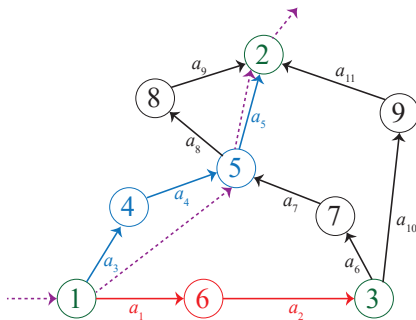
- Node memory
- Arc memory

# Node memory



$S = \{1, 2, 3\}$  and node memory  $M = \{1, 2, 3, \dots, 9\}$

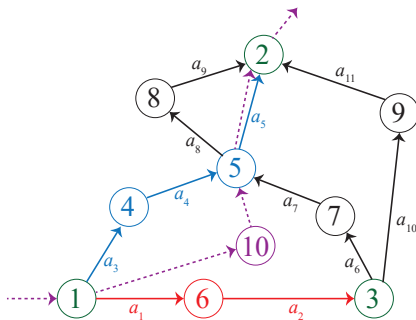
# Node memory



$S = \{1, 2, 3\}$  and node memory  $M = \{1, 2, 3, \dots, 9\}$

Path 1-5-2 stays in the memory between 1 and 2 and contributes to the cut

# Node memory



$S = \{1, 2, 3\}$  and node memory  $M = \{1, 2, 3, \dots, 9\}$

Path 1-10-5-2 leaves the memory between 1 and 2 and **does not contribute** to the cut

# Variable fixing

Based on the following result

Let  $LB(\pi)$  be a lower bound derived from a feasible dual solution  $\pi$  to the MP. Let  $UB$  be an upper bound.

If  $\bar{c}_p(\pi) > UB - LB(\pi)$ , then  $\theta_p = 0$  in any optimal solution

## Procedure

- 1 Given an upper bound  $UB$  (e.g., from a heuristic solution)
- 2 Compute a lower bound  $LB(\pi)$  and a dual solution  $\pi$  (e.g., by column generation with cuts)
- 3 Solve the PP twice: forwardly and backwardly
- 4 For each arc  $(i, j)$ , compute the least reduced cost  $\hat{c}_{ij}(\pi)$  of all routes using this arc
- 5 If  $\hat{c}_{ij}(\pi) > UB - LB(\pi)$ , remove arc  $(i, j)$  (i.e., all routes containing  $(i, j)$ )

# Variable fixing

Based on the following result

Let  $LB(\pi)$  be a lower bound derived from a feasible dual solution  $\pi$  to the MP. Let  $UB$  be an upper bound.

If  $\bar{c}_p(\pi) > UB - LB(\pi)$ , then  $\theta_p = 0$  in any optimal solution

## Procedure

- ➊ Given an upper bound  $UB$  (e.g., from a heuristic solution)
- ➋ Compute a lower bound  $LB(\pi)$  and a dual solution  $\pi$  (e.g., by column generation with cuts)
- ➌ Solve the PP twice: forwardly and backwardly
- ➍ For each arc  $(i, j)$ , compute the least reduced cost  $\hat{c}_{ij}(\pi)$  of all routes using this arc
- ➎ If  $\hat{c}_{ij}(\pi) > UB - LB(\pi)$ , **remove arc  $(i, j)$  (i.e., all routes containing  $(i, j)$ )**

# Route enumeration

Based on the following result

Let  $LB(\pi)$  be a lower bound derived from a feasible dual solution  $\pi$  to the MP. Let  $UB$  be an upper bound.

If  $\bar{c}_p(\pi) > UB - LB(\pi)$ , then  $\theta_p = 0$  in any optimal solution

A route enumeration algorithm

- 1 Compute an upper bound  $UB$  (e.g., using a heuristic)
- 2 Compute a lower bound  $LB(\pi)$  and a dual solution  $\pi$  (e.g., by column generation with cuts)
- 3 Enumerate (by dynamic programming) all routes  $p$  with  $\bar{c}_p(\pi) \leq UB - LB(\pi)$
- 4 Solve a MIP considering only these variables

# Route enumeration

## Based on the following result

Let  $LB(\pi)$  be a lower bound derived from a feasible dual solution  $\pi$  to the MP. Let  $UB$  be an upper bound.

If  $\bar{c}_p(\pi) > UB - LB(\pi)$ , then  $\theta_p = 0$  in any optimal solution

## A route enumeration algorithm

- 1 Compute an upper bound  $UB$  (e.g., using a heuristic)
- 2 Compute a lower bound  $LB(\pi)$  and a dual solution  $\pi$  (e.g., by column generation with cuts)
- 3 **Enumerate** (by dynamic programming) **all routes**  $p$  with  $\bar{c}_p(\pi) \leq UB - LB(\pi)$
- 4 **Solve a MIP** considering only these variables

# Improvements

## Combined with branching

- If too many routes are to be enumerated, **stop enumeration and branch**
- Apply route enumeration at every node of the search tree

## Column pool

- Allow the enumeration of a large number of routes
- Store them in a **column pool**
- Add as many cuts as possible to raise the lower bound
- Use **column generation** to solve the relaxation, where the PP consists of **pricing the columns in the pool**
- Perform additional variable fixing and repeat
- If the number of routes is sufficiently small, solve a MIP

# Improvements

## Combined with branching

- If too many routes are to be enumerated, **stop enumeration and branch**
- Apply route enumeration at every node of the search tree

## Column pool

- Allow the enumeration of a large number of routes
- Store them in a **column pool**
- Add as many cuts as possible to raise the lower bound
- Use **column generation** to solve the relaxation, where the PP consists of **pricing the columns in the pool**
- Perform additional variable fixing and repeat
- If the number of routes is sufficiently small, solve a MIP

# Outline

- 1 VRPTW definition
- 2 Mathematical formulations
  - Arc-flow formulation
  - Dantzig-Wolfe decomposition
  - Path-flow formulation
- 3 Branch-price-and-cut
  - Master problem
  - Pricing problem
  - Linear relaxation computational results
  - Branching
  - Cutting
  - Using upper bounds
- 4 Computational results (VRPTW and CVRP)

## VRPTW

Class	100 customers			200 customers		
	#Inst	#Opt	Time (s)	#Inst	#Opt	Time (s)
C1	9	9	15	10	10	133
RC1	8	8	52	10	8	38940
R1	12	12	31	10	10	12854
C2	8	8	328	10	8	4096
RC2	8	8	337	10	7	27375
R2	11	11	6432	10	8	13645
Total	56	56		60	51	
Average			1375			15195
Max			64105	.		188414

From Pecin et al., 2017a

# Capacitated VRP

Class	#Inst	#Cust		Opt	Time (s)	
		Min	Max		Avg	Max
A	22	37	80	22	2	8
B	20	38	78	20	7	98
E-M	12	51	200	12	2712	28620
F	3	45	135	3	3183	4953
P	24	16	101	24	21	336
C	7	50	199	7	3341	22090
G	7	240	360	4	886381	3354659

Largest instance solved with 360 customers!

From Pecin et al., 2017b

# A few references 1

- R. Baldacci, N. Christofides, A. Mingozzi (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming* 115(2), 351–385.
- R. Baldacci, A. Mingozzi, R. Roberti (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research* 59(5), 1263–1283.
- N. Boland, J. Dethridge, I. Dumitrescu (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters* 34, 58–68.
- C. Contardo, J.-F. Cordeau, B. Gendron (2014). An exact algorithm based on cut and column generation for the capacitated location-routing problem. *INFORMS Journal on Computing* 26(1), 88–102.
- C. Contardo, R. Martinelli (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization* 12, 129–146.
- G. Desaulniers, F. Lessard, A. Hadjar (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science* 42(3), 387–404.
- Desrochers, M., J. Desrosiers, and M.M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.
- Feillet, D., P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.
- S. Irnich, G. Desaulniers, J. Desrosiers, A. Hadjar (2010). Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing* 22(2), 297–313.

## A few references 2

- M. Jepsen, B. Petersen, S. Spoorendonk, D. Pisinger (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research* 56(2), 497–511.
- Kohl, N., J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis. 2-Path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999.
- R. Martinelli, D. Pecin, M. Poggi (2014). Efficient elementary and restricted non-elementary route pricing. *European Journal of Operational Research* 239(1), 102–111.
- D. Pecin, C. Contardo, G. Desaulniers, E. Uchoa (2017a). New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing* 29(3), 489–502.
- D. Pecin, A. Pessoa, M. Poggi, E. Uchoa (2017b). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation* 9(1), 61–100.
- A. Pessoa, E. Uchoa, M. Poggi de Aragão (2009). A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Networks* 54(4), 167–177.
- G. Righini, M. Salani (2006). Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization* 3(3), 255–273.
- G. Righini, M. Salani (2008). New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks* 51(3), 155–209.
- C. Tilk, A.-K. Rothenbächer, T. Gschwind, S. Irnich (2017). Asymmetry helps: Dynamic half-way points for solving shortest path problems with resource constraints faster. *European Journal of Operational Research* 261(2), 530–539.